COURSERA FINAL PROJECT

SPECIALIZED MODELS: TIME SERIES AND SURVIVAL ANALYSIS

Time Series Forecasting to predict grocery sales at Favorita stores

Project presented by

Raziel Amador Ríos, Ph.D.

to obtain the Coursera Certificate

Main object: produce a reliable forecast based on sales time series data from a retail store (Favorita stores, an Ecuadorian-based company).

Barcelona, February 2023

Table of Contents

Time Series Forecasting

I

Main objective

The **main object** of the project is to:

- Generate a reliable **time series forecast** based on sales from Favorita stores.

Favorita stores is a grocery retail store based in Ecuador. Producing an accurate forecast could lead to decreased food waste related to overstocking and improve customer satisfaction. All the code can be found in the following GitHub repository, further information can be found in Additional information.

II

Data description

The dataset comes from **Kaggle**, named: Store Sales - Time Series Forecasting. The Kaggle dataset contains sales data from *Corporación Favorita*, a large Ecuadorian-based grocery retailer. Kaggle provides you with 7 diffeent files (see Table 1 for more details).

Number	File Name	Description
1	holiday_events.csv	Relevant holidays in Ecuador
2	oil.csv	Oil prices from 2013 to 2017
3	sample_submission.csv	submission example
4	stores.csv	Stores metadata
5	test.csv	Stores and family products
6	train.csv	Sales by store and product-family from 2013-01 to 2017-08
7	transactions.csv	Number of transactions by store

Table 1: Kaggle files description. Files are alphabetically sorted.

The training data represents 99% of the data, including dates from 2013-01-01 to 2017-08-16 (55.5 months), 54 stores placed in different cities within Ecuador, and 33 family-products (see Figure 1). The testing data includes dates from 2017-08-16 to 2017-08-31 (15 days).

Figure 1: Summary of the training dataset. The cluster information denotes similarity between stores.

III

Data exploration and data cleaning

Performing an exploratory data analysis (EDA) of the sales from the training dataset, we can observe that grocery I, beverages, and produce are the top 3 most consumed products (see Figure 2) . Additionally, store type A, and cluster 5 are the most frequent among their classification (Figure 2).

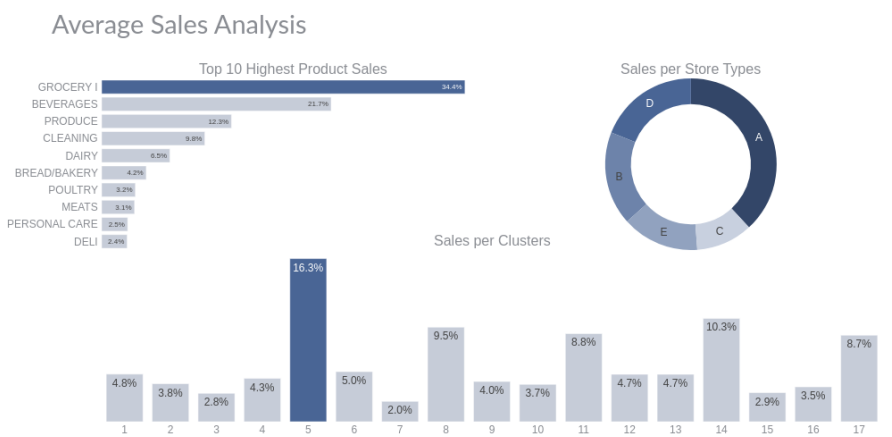

Figure 2: EDA of sales. The plot describes the sales by product, store type, and per cluster (left, right, and below, respectively). Darker blue represents higher sales.

The sales represented by Figure 3 shows low-peaks at the end of each year, which is explained because the stores are closed at New years. Moreover, we can observe a pattern at each year, with increased sales at the end of the year which overlaps with the Christmas eves, suggesting a seasonal pattern. These patterns are highlighted after smoothing the sales data with a 7 days moving average (the continuous red lines from the Figure 3).

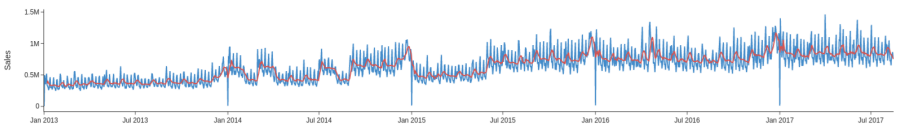

Figure 3: Target time series data. The y-axis represents the training sales from 2013-01-01 to 2017-08-15, and the x-axis shows the time in days (1087 days). The sales were aggregated by all stores and products (see Figure 1). The raw data, and smoothed data (7 days moving average) are denoted by the continuous blue, and red lines, respectively.

III.1. Time series Decomposition

Time series are defined as a sequence of data organized in time order. The key components of time-series are: trend, seasonality, and residuals, displayed in Figure 4B, Figure 4C, and Figure 4D, respectively. Trend is the long-term direction of the time series, seasonality describes the periodic behavior such as holidays, and residuals represent the irregular fluctuations that we are not able to predict using the trend and/or seasonality.[1]

In our target time series, we observe an upward trend, with a clear seasonality factor (this will be further analyzed below), and the residuals do not follow a random pattern. These insights will be used to select an adequate forecast model.

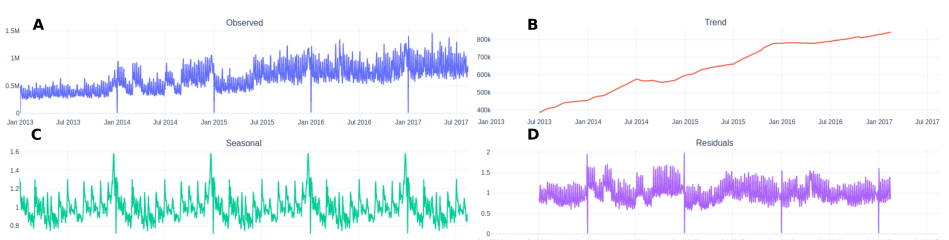

Figure 4: Time series decomposition. **(A)** Raw sales time series (sames as Figure 3). **(B)** Trend of sales. **(C)** Seasonality of sales. **(D)** Residuals of sales.

III.2. Stationarity and seasonality analyses

The stationary and seasonality are relevant components to select the adequate machine learning model (such as ARIMA, SARIMA, etc.) to generate a reliable forecast. A stationary time series is defined, if their statistical properties such as mean, and variance are all constant, and independent of time.[1,2] In consequence, we implemented the *Dickey-Fuller test* to assess stationarity. We obtained a *p-value* of 0.09, using a significance level of 0.05, we can reject the null hypothesis (the series is stationary) concluding that our series is non-stationarity.

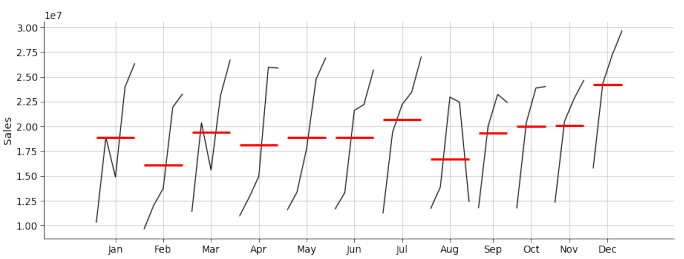

Figure 5: Analysis of seasonality. The y-axis represents the sales aggregated by month, x-axis shows the analyzed months (from January to December), the horizontal red lines denote the sales average by month (55.5 months).

In terms of seasonality, we aggregate the sales for each month and we can clearly see on average an increase of sales during December (Figure 5), which makes sense because Christmas and New years represent a season of the year with more transactions in other retail businesses. Furthermore, we highlighted February as the lowest sales month (Figure 5).

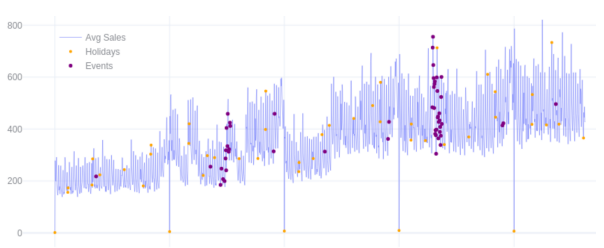

Figure 6: Holidays effect. Holidays and special events are represented as yellow, and purple dots, respectively.

Holidays are one of the most important factors for seasonality, and the Kaggle dataset provided us with relevant holidays and special events in Ecuador (see Table 1). Thus, we analyzed the effects of holidays on sales (see Figure 6). We observed a reasonable correlation between holidays and sales. Consequently, holidays must be incorporated on the forecast.

IV

Time series Models

The main goal of the project is to generate a reliable forecast using the sales data from Favorita stores. After the EDA, we concluded that our data is not stationarity, possesses a strong seasonal effect, and holidays is a factor that should be considered in the forecast. Therefore, the following models could be implemented:

- **SARIMAX**[3]: Seasonal ARIMA using eXogenous data. Further, ARIMA stands for Auto-Regressive (p: dependence on past values), Integrated (d: differencing) Moving Average (q: dependence on past forecast errors).
- **LSTM**[4]: Long-Short Term Memory.
- **GRU**[5]: Gated Recurrent Unit.
- **Facebook Prophet**[6]: is a forecasting procedure similar to a generalized additive model[7] (GAM) where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.
- **NeuralProphet**[8]: is a hybrid forecasting framework based on PyTorch and trained with standard deep learning methods.
- **LightGBM**[9] (or other extreme gradient boosted method, *e.g.* XGBoost): Light Gradient-Boosting Machine.

Ideally, we would like to implement all of the machine learning models mentioned above, do a benchmark taking into account: performance, computational-resources, training-time, and model-explainability; and compare their results. Nonetheless, due to time-constraints restrictions, we selected **Facebook Prophet** as our unique machine learning model to forecast the sales from Favorita stores. The reasons to select Facebook Prophet include:

1. A simple and flexible model (see Equation 1 and Equation 8) that takes into account multiple human scale seasonality, and holidays that occur at irregular intervals.

2. Short training time (using the *L-BFGS* optimization algorithm[10]).

3. Low computation-resource demands and a Python module to easily implement the model (under the hood uses Stan[11]).

4. Model explainability.

5. Previous experience with the model.

IV.1. Prophet as our Forecast model

According to Taylor *et al.*[6], Facebook Prophet or Prophet (we are going to use Prophet from now on) works best with time series that have strong seasonal effects (which is our case see Figure 4C, and Figure 5), holidays effect (Figure 6), and typically handles outliers well. The model contains three main components: **1)** trend, **2)** seasonality, and **3)** holidays, which are combined in Equation 1:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{1}$$

Where $g(t)$ is the trend function which models non-periodic changes in the value of the time series, $s(t)$ represents seasonality (*e.g.*, weekly and yearly seasonality), and $h(t)$ represents the effects of holidays which occur on potentially irregular schedules over one or more days. The error term ε_t represents changes not explained by the model; default prior $\varepsilon \sim N(0, 0.5)$.

Prophets implements two types of trend models (logistic and linear trend model). In our project, the linear trend with changepoints was used, and can be written as Equation 2. Where κ stands for the growth rate, δ has the rate adjustments, m is the offset parameter, and γ_j is set to $-s_j\delta_j$ to make the function continuous. Additionally, prior $\delta_j \sim Laplace(0, \tau)$ where τ regularizes the trend flexibility.

$$g(t) = (\kappa + a(t)^T\delta)t + (m + a(t)^T\gamma) \tag{2}$$

Next, seasonality $s(t)$ is modeled based on the Fourier series Equation 3. Where P represents the regular period we expect the time series to have (*e.g.* $P = 365.25$ and $P = 7$ for yearly and weekly data, respectively). The term $X(t)\beta$ can be further explained with $X(t)$ and β specified as vectors, see Equation 4 and Equation 5. The term N represents the order of the Fourier series. For yearly and weekly seasonality, the authors have found $N = 10$ and $N = 3$ to work well for most series problems, respectively. According to Taylor *et al.*, prior $\beta \sim N(0, \sigma^2)$ where σ regularizes the strength of seasonality.

$$s(t) = \sum_{n=1}^{N} \left(a_n cos\left(\frac{2\pi nt}{P}\right) + b_n sin\left(\frac{2\pi nt}{P}\right)\right) = X(t)\beta \tag{3}$$

$$X(t) = \left[cos\left(\frac{2\pi 1t}{P}\right), sin\left(\frac{2\pi 1t}{P}\right), ..cos\left(\frac{2\pi Nt}{P}\right), sin\left(\frac{2\pi Nt}{P}\right)\right] \tag{4}$$

$$\beta = [a_1, b_1, ...a_N, b_N] \tag{5}$$

Subsequently, Equation 6 represents the linear effects of holidays, and can be extended as Equation 7 which is a vector of dummies (1 indicates holiday and 0 otherwise), by assuming that the effects of holidays are independent. For each holiday i, let D_i be the set of past and future dates for that holiday. And assign each holiday a parameter κ_i which is the corresponding change in the forecast. As before, we use a prior $\kappa \sim N(0, \nu^2)$ where ν^2 regularizes the holiday effects.

$$h(t) = Z(t)\kappa \tag{6}$$

$$Z(t) = [1(t \in D_1), ..., 1(t \in D_L)] \tag{7}$$

Finally, combining all three models described above, we obtain the final model for our time series analysis that can be written as Equation 8. Then, based on the five priors of the parameters and the data, we can find maximum a posterior (MAP) estimates for all parameters.

$$y|m, \delta, \beta, \kappa, \varepsilon \sim N(g(t) + s(t) + h(t), \varepsilon) \tag{8}$$

IV.1.1. Prophet Hyperparameters

Prophet possesses 4 hyperparameters which can be tuned through cross validation, and are the following:

1. *changepoint prior scale*: Determines the flexibility of the trend, in particular the trend changes, and the trend changepoints. If the value of changepoint prior scale is too small leads to underfit, if too large leads to overfit and in extreme scenarios leads with the trend capturing yearly seasonality. Further, higher values means more changepoints and more flexible trend.

2. *seasonality prior scale*: Large values allow the seasonality to fit large fluctuations, and small values shrink the magnitude of the seasonality. The default value (see Table 2) applies basically no regularization that is because the authors rarely see overfitting here.

3. *holidays prior scale*: This controls flexibility to fit holiday effects. Likewise to *seasonality prior scale* the default value applies no regularization, since we usually have multiple holiday observations and can do a good job of estimating their effects.

4. *seasonality model*: This term, when possible, is best identified graphically by observing if the magnitude of seasonal fluctuations grows with the time series (multiplicative). Nonetheless, when it is not possible, it could be tuned.

Hyperparameter	Default	Recommended Range	Term
changepoint prior scale	0.05	[0.001, 0.5]	τ
seasonality prior scale	10	[0.01, 10]	σ
holidays prior scale	10	[0.01, 10]	ν
seasonality model	additive	additive or multiplicative	$g(t) + s(t)$ or $g(t) * s(t)$

Table 2: Prophet Hyperparameters. The equation terms are defined above, see Prophet as our Forecast model.

IV.1.2. Prophet Cross validation

In our work, only the *seasonality mode* (either additive or multiplicative) was tuned using cross validation (CV). CV in Prophet uses historical data and compares the forecasted values with the real values. There are three parameters we need to define:

1. *initial*: the size of the initial training period.
2. *period*: space between two or more training periods, *i.e.* cutoff dates.
3. *horizon*: forecast horizon, *i.e.* how many days/weeks,months,years are we going to make a forecast on.

Since we have 1087 days we are using the initial 730 days (67.16%) to predict the next 365 days, although all the results and plots presented show the performance metric (see Performance Metric) of a half year horizon. The period used was 180 days. Table 3 presents a more detailed description of the used parameters in the cross validation function.

Parameter	Default	Selected
initial	3*horizon	730 days (67.16%)
period	0.5*horizon: 182 days	180 days
horizon		365 days

Table 3: Cross validation in Prophet. The selected column displays the implemented parameters across all forecast results.

IV.2. Performance Metric

To assess the performance of our models, we selected the mean absolute percentage error (MAPE) which is defined below:

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

Where A_t is the actual value (true value) and F_t is the forecast result (predicted value). MAPE was selected as a performance metric because of its simplicity to represent the forecast errors within a percentage scale, where the lower the better.

V

Forecast Results

The sales data can be organized by store (54 stores) and by store and family product (1782 combinations). Therefore, as a starting point we aggregate all sales and generate 1 forecast (see General Sales Forecast). Then, generate a forecast by each store (54 forecasts; see Sales Forecast by Store). Finally, we generate a forecast of each product within each store (1782 forecasts; see Sales Forecast by Store and Product).

V.1. General Sales Forecast

Figure 7 shows the forecast outcome obtained aggregating all the sales, obtaining a MAPE of 22.26%.

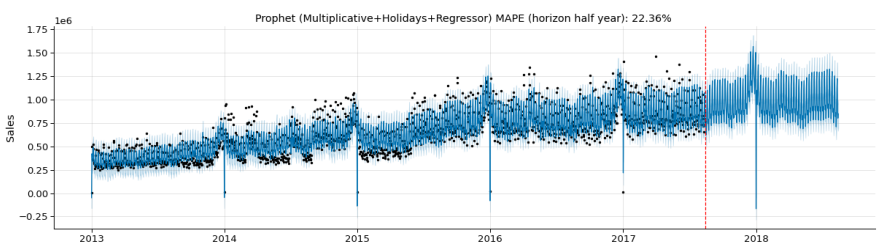

Figure 7: General Sales Forecast. Blue lines denotes the forecast, and black dots describe the actual values. Forecast is represented after the red dashed line.

After experimenting with different hyperparameters and conditions, the Prophet model with the highest performance (the lowest MAPE) considers the following:

1. A multiplicative seasonality mode.
2. Uses the provided holidays (holiday_events.csv).

3. Custom regressor representing the store closing days.

We observe a correct modeling of the training data, and correctly modeling the pronounced dips around Christmas and New Year (Figure 7). Model explainability is an important factor to select a final model, in addition to performance, and computational training time. Prophet comes with an integrated explanation of the model components including the effects of holidays, weekly and yearly seasonality which helps to assess the performance of the forecast.

In consequence, the forecast components are displayed in Figure 8. A sales increase is reported on the holiday component (Figure 8A), which is expected in the retail business. In terms of weekly and yearly seasonality, Saturday and Sunday; and December are the periods with the highest sales, respectively (Figure 8B and C).

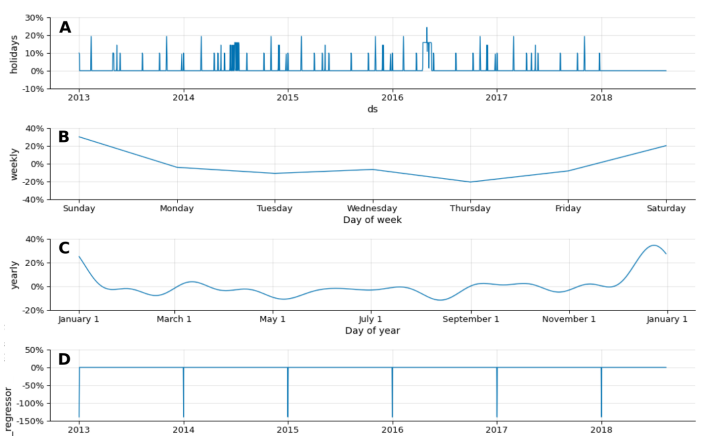

Figure 8: General Forecast Components. The components of the final Prophet model, which considers a multiplicative seasonality, holidays, and a custom regressor represented on Figure 7. **(A)** The holiday component. **(B)** Weekly seasonality. **(C)** Yearly seasonality. **(D)** Regressor considering the store closing days.

The regressor displays the store closing days which explains the pronounced dips around Christmas and New Year (Figure 8D). In conclusion, we can state that our proposed models possess a moderate error (MAPE of 22.26%), and the forecast components are in alignment of what we would expect of the retail business.

V.2. Sales Forecast by Store

Our dataset contains 54 stores. Thus, one time series model was produced to each store leading to 54 different Prophet models. In this scenario, a vanilla Prophet model was fitted to each store. Obtaining a mean MAPE of 36.87%, twelve random stores are represented by Figure 9, where the orange lines show the sales forecast by store. For better figure representation, the stores are displayed from 2017-05 to 2017-09 to highlight the last dates.

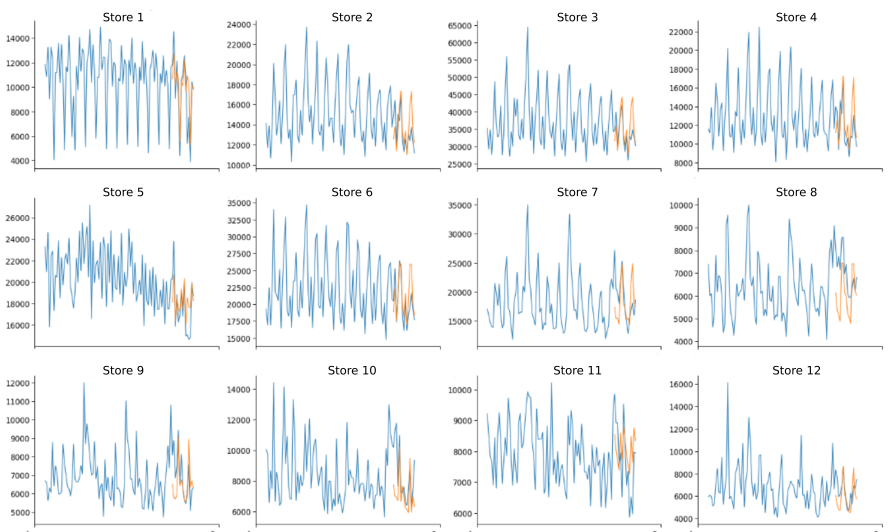

Figure 9: Forecast by Store. The x-axis represents the dates from 2017-05 to 2017-09, the y-axis shows the sales for each store. The blue and orange lines display the store sales and forecast, respectively.

As expected, we obtained a higher error compared to the general sales forecast (36.87% vs. 22.26%). The reasons are: each store has a different sales pattern, and each Prophet model should be tuned for each store. In addition, a data normalization (such as *log* transformation) could be implemented to achieve a lower MAPE. Nonetheless some store predictions are reasonable, for instance for stores: 1, 3, 6, and 10 (Figure 9). In contrast, for stores: 7, 8, and 11 the forecasts can be considered as non-correct (Figure 9).

Overall, 88% of the forecast can be considered as correct with a mean MAPE of 25.96% (see all the forecast plots at the GitHub repository). Considering the complexity of time series forecasting and the different sales patterns for each store, we can

label our results as satisfactory. In order to have a higher confidence on our predictions a visual inspection should be used before relying on our forecast predictions.

V.3. Sales Forecast by Store and Product

In this scenario, one vanilla Prophet model was fitted for each product for each store producing a total of 1782 forecasts. As we can see on Figure 10, each product for each store presents a completely different sales behavior leading to a more complex forecast problem. On average, we obtained a MAPE of 61.02% (see Table 4). Consequently, we are not able to suggest to rely on our predictions following this approach.

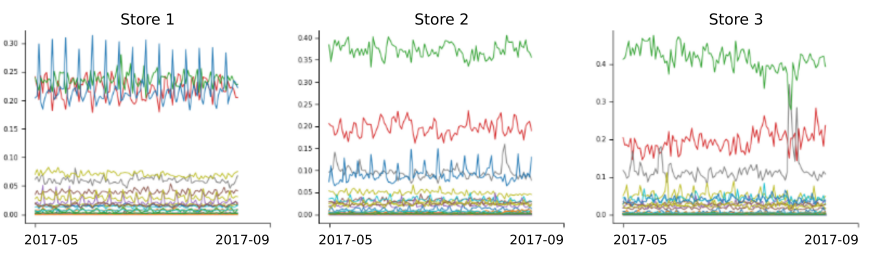

Figure 10: Forecast by Store and Product. The x-axis represents the dates from 2017-05 to 2017-09, the y-axis shows the sales for each store and product. The different color lines represent each sales product.

As Figure 10 shows each product type presents a different pattern on each store, explaining the low performance of our vanilla Prophet models, the same observation is maintained for all the 54 stores.

Forecast Approach	MAPE	Number of Predictions
General Sales Forecast	22.26%	1
Sales Forecast by Store	36.87%	54
Sales Forecast by Store and Product	61.02%	1782

Table 4: Forecast Summary Results. Results are sorted by MAPE.

In order to improve the predictions performance, we need to tune each model following the same approach of the General Sales Forecast (hyperparameter tuning, incorporating the holidays, etc.), but due to computational resources and time constraints these improvements were not implemented. Further improvements will be discussed in the Next Steps section.

VI

Conclusions

VI.1. Recommendations

After performing general, store-specific, and product-specific forecasts we can recommend incorporating our general and store-specific forecast predictions, the latter after a visual inspection. Our general and store-specific predictions present a reasonable error, and their explainability components are in alignment of what we would expect of the retail business (see Figure 8 and Table 4, respectively).

Moreover, the training time of the models is not a constraint, and we could decrease the prediction horizon to decrease the MAPE metric, and retrain the models each month to avoid model drifting. Additional improvements in our general and store-specific will be discussed in the Next Steps section. Generating a reliable forecast for each product within each store could be a great interest for any retail business. Nevertheless, our results show on average a high error rate (see Table 4) and we cannot recommend using our predictions following the product-specific approach.

VI.2. Summary Key Findings

VI.3. Next Steps

Bibliography

1. Moroff, N. U., Kurt, E. & Kamphues, J. Machine Learning and statistics: A Study for assessing innovative demand forecasting models. *Procedia Computer Science* **180,** 40–49 (2021).
2. Feizabadi, J. Machine learning demand forecasting and supply chain performance. *International Journal of Logistics Research and Applications* **25,** 119–142 (2022).
3. Hyndman, R. J., Koehler, A. B., Snyder, R. D. & Grose, S. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting* **18,** 439–454 (2002).
4. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9,** 1735–1780 (1997).
5. Cho, K. *et al.* Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
6. Taylor, S. J. & Letham, B. Forecasting at scale. *The American Statistician* **72,** 37–45 (2018).
7. Hastie, T. & Tibshirani, R. Generalized additive models: some applications. *Journal of the American Statistical Association* **82,** 371–386 (1987).
8. Triebe, O. *et al.* Neuralprophet: Explainable forecasting at scale. *arXiv preprint arXiv:2111.15397* (2021).
9. Ke, G. *et al.* Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* **30** (2017).
10. Byrd, R. H., Lu, P., Nocedal, J. & Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing* **16,** 1190–1208 (1995).
11. Carpenter, B. *et al.* Stan: A probabilistic programming language. *Journal of statistical software* **76** (2017).

Appendix

VII

Additional information

This work was written with emacs[1] using LaTeX[2], using only **Free and Open Source software**. All the computational analysis were carried out using Linux-based distributions. The figures were generated with Python (matplotlib[3]/seaborn[4]/plotly[5]) and Inkscape[6].

Contact: razielar@gmail.com

[1] https://www.gnu.org/software/emacs/
[2] https://www.latex-project.org/
[3] https://matplotlib.org/
[4] https://seaborn.pydata.org/
[5] https://plotly.com/
[6] https://inkscape.org/